Comprehensive Summary and Analysis of "Privadome: Delivery Drones and Citizen Privacy"

1. Introduction: The Privadome System

The research paper **"Privadome: Delivery Drones and Citizen Privacy"** introduces a system designed to protect the privacy of citizens from the cameras on autonomous delivery drones. The system empowers citizens by giving them tools to verify that their privacy is being respected.
The system is comprised of two main components:

- **Pd-Mpc:** Allows a citizen to securely identify which specific drones in their vicinity may have captured their footage, without revealing their own location.
- **Pd-Ros:** Allows the citizen to communicate with an identified drone to obtain a secure audit trail, proving that their footage was handled according to privacy-preserving rules.

This report focuses on the detailed explanation of **Pd-Ros**, as covered in Section 4 of the paper.

2. Auditing Drone Compliance with Pd-Ros

The primary goal of Pd-Ros is to provide citizens with verifiable proof that a "well-intentioned" drone is protecting their privacy. It aims to achieve two goals:

- **(G1) Use only sanitized data:** The drone must ensure that raw camera data is never used directly. It must first be "sanitized" (e.g., by blurring faces).
- **(G2) Offer compliance proof:** The drone must be able to provide undeniable, cryptographic proof to a citizen that Goal G1 is being met.

To achieve this, the system relies on **trusted hardware**, specifically **ARM TrustZone**. This technology divides the drone's processor into two isolated environments:

- **Normal World (Untrusted):** Where the drone's main operating system (Linux) and complex robotics software (ROS2) run.
- **Secure World (Trusted):** A small, tamper-proof environment that runs only critical, security-verified code.

3. The Technical Foundation (ROS2 & SROS)

The drone's software is built on **ROS2 (Robot Operating System 2)**, a standard framework for robotics. ROS2 uses a flexible **publish/subscribe** model where different software components (e.g., camera, navigation) can communicate by sending messages to shared channels, known as "topics."
However, standard ROS2 is insecure. To fix this, the Privadome system is built on **SROS**

(Secure ROS), which adds critical security layers:

- **Encryption:** It uses TLS (the same tech as HTTPS) to encrypt all messages, preventing eavesdropping.
- **Strict Permissions:** Every application must have a digitally signed **manifest**—a permission slip that explicitly lists which topics it is allowed to access. SROS enforces these rules strictly.

4. System Architecture and Data Flow

Pd-Ros enforces privacy by creating a mandatory, secure detour for all video data, policed by the Secure World.

1. **Capture:** The camera app publishes raw video to a private topic called VideoFeed. No normal drone application is allowed to access this topic.
2. **Redirection:** The feed is exclusively sent to a gatekeeper app called the ① **Sanitizer Front-end (FE)**.
3. **Sanitization:** The Sanitizer-FE immediately passes the raw video to the ② **Trusted Video Sanitizer** in the Secure World. This trusted component applies privacy filters (like blurring) and returns the clean video.
4. **Publication:** The Sanitizer-FE publishes this clean, "sanitized" video to a public topic called PrivVideoFeed, which all other drone apps can safely use.

This flow ensures **Goal G1** is met. To achieve **Goal G2**, the ③ **App Launcher** collects the signed manifests of all running apps and sends them to the ④ **Trusted Audit Trail Logger** in the Secure World. This logger stores them as tamper-proof evidence.

5. The Citizen's Verification Process

When a citizen requests proof, the Trusted Audit Trail Logger sends them the digitally signed collection of manifests. The citizen then performs a 3-step check:

1. **Check the Foundation:** Verify the attestation report to ensure the drone's core OS and SROS software have not been compromised.
2. **Check the Gatekeepers:** Verify the integrity of the critical Sanitizer-FE and App Launcher applications to ensure they haven't been swapped with malicious versions.
3. **Check the Rules:** Inspect the manifests to confirm that only the Sanitizer-FE accessed the raw VideoFeed and that all other apps correctly used the sanitized PrivVideoFeed.

This process provides verifiable proof of compliance. The paper notes that these queries must be made anonymously (e.g., via **Tor**) to protect the citizen's location privacy. The key advantage over a centralized reporting system is that this method provides **personal accountability**, allowing a citizen to check the specific footage that may involve them, akin to verifying a blurred house on Google Street View.

6. Performance Evaluation and Trade-offs

The researchers tested the system on an **Nvidia Xavier NX** board, which is similar to hardware on real drones. They measured the performance cost of their security system.

- **Real-Time Performance:** The secure video redirection and blurring process added almost no time delay **(+0.26% latency)**, meaning it works in near real-time. However, this came at the cost of significantly higher resource consumption: **+70.34% CPU utilization** and **+101.8% power draw** for the CPU/GPU.

- **Startup Performance:** The upfront security checks (attestation and logging manifests) more than doubled the time it takes to launch an application, from **4.78 seconds to 10.66 seconds**.

Conclusion: The evaluation demonstrates that Pd-Ros is a feasible system. The privacy protections come with clear and measurable trade-offs: a minimal impact on real-time operations but a significant increase in power consumption and a one-time delay at startup.